Каждое событие в фрэймворке (в программе) записывается, и эта запись и называется логами, в формате текста в файл. Для автоматизации это полезно тем что если мы воспроизводим 100 тестов и какойто тест зафэйлился то в логах можно будет оперативно отследить когда, где, что и почему упало.

Так, походу бывают аутомэйшн логс и апликейшн логс.

Логи подразделяются по уровням, когда указываешь уровень, автоматически в логи будут включены все ниже идущие уровни.

All < Trace < Debug < Info < Warn < Error < Fatal < Off

**Debug** - it will capture < Debug < Info < Warn < Error < Fatal, it is able to catch all the logs (application + automation + communication between client and server). This log level requires a beeg size of a file, because a lot of info is there.

**Warn** - it will capture < Warn < Error < Fatal. in that scenario we are able to proceed further.

**Error** - it will capture < Error < Fatal. if you want to capture errors this is the level.

**Fatal** - if somewhere something got totally blocked and we are not able to proceed further.

Логи входят в баг репорт как правило это уровень Debug.

**Чтобы использовать логи нужна сторонняя библиотека:**

**<dependency>**

**<groupId>org.apache.logging.log4j</groupId>**

**<artifactId>log4j-api</artifactId>**

**<version>2.23.1</version>**

**</dependency>**

**<dependency>**

**<groupId>org.apache.logging.log4j</groupId>**

**<artifactId>log4j-core</artifactId>**

**<version>2.23.1</version>**

**</dependency>**

Теперь нужен документ **log4j2.xml**. Это конфигурационный файл в нём мы можем обозначить/указать какой уровень логов нам нужно сгенерировать, где они должны быть сгенерированы (можно в файле, можно в консоли). log4j2.xml файл должен быть расположен в **src.test.resources** пакете, так как этот файл является ресурсом.

Есть сайт Apache Log4j там есть пример как составить log4j2.xml файл

Пример:

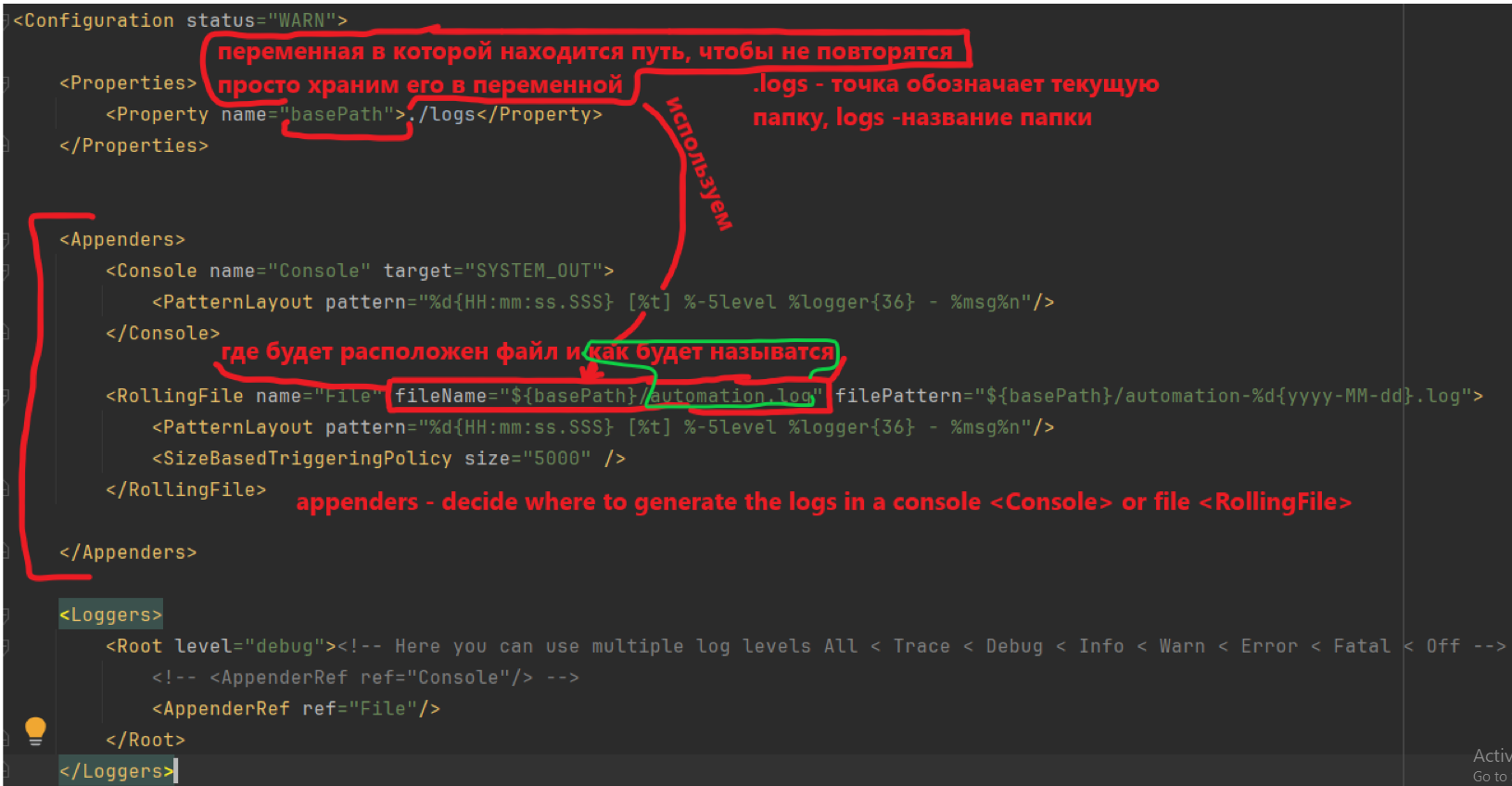
Сам документ имеет 2 большие части:

<appender> - как и где генерить логи

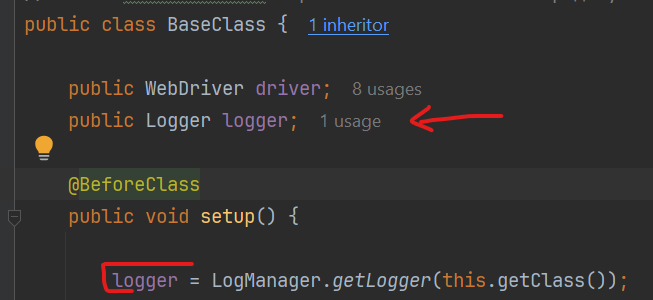
<Loggers> - какой тип генерить и где окончательно

В appender (здесь решается где будут генерироватся логи в консоли или в файле) настроены логи которые будут отражаться и в консоли и записыватся в файл. Так как детали из консоли доступны только в ран тайме, а файл можно просмотреть или отправить в баг рипорте.

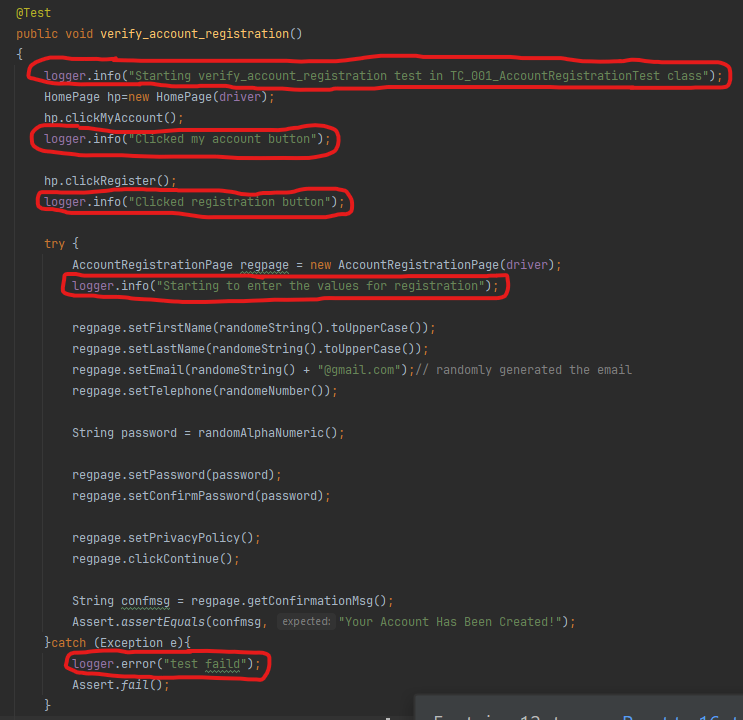
В <Loggers> мы указываем уровень на котором хотим отслеживать логи. При указании уровня надо помнить что если указал уровень например info то можеш работать в коде с этим уровнем и всё что ниже ( Info < Warn < Error < Fatal) , но выше уровень (например Debug) ты обработать не сможешь. Для этого есть тег <Root> и атрибут level куда и надо подставить значение уровня. <AppenderRef> и атрибут ref обозначает что мы записываем в файл, но можно вместо значения ‘файл’ указать консоль тогда логи будут генерироватся в консоли. Здесь идёт привязка к тому что указано в аппендерах. Мне кажется это как быстрый переключатель, хочеш генерим в консоль, хочеш в файл, а настройки все выполнены в аппендерах.



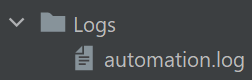
После, надо интегрировать логи с тестами. Для этого есть класс Logger, его имплементирует какойто LogManager с помощью статического метода getLogger(название\_класса\_который\_будет\_использовать\_Logger), здесь в примере класс в котором имплементируется Logger является прародителем всех классов тест кейсов которые унаследуют этот Logger и смогут его использовать. Т.е. класс BasePage что-то типа утилиты, поэтому и пишем this.



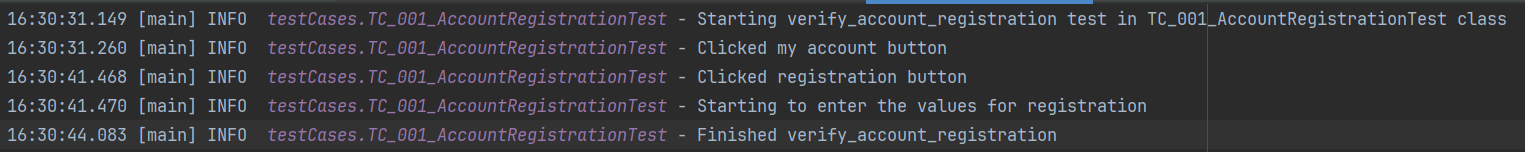
Теперь надо использовать объект logger который мы создали у него есть разные методы, (info, error…) основная суть расставить в тесте логи, где надо используй info, если возможна ошибка то обойди try catch-ем и словив исключение запиши логом ошибку т.е. error (помни про уровни выбранные в конфигурационном файле, ты не можеш обработать логи на уровне Debug если у тебя выбран уровень info), смотри скрин ниже, логи будут после рантайма указаны в соответствующем документе в указанной папке



После этого будет создана папка там где было указано в **log4j2.xml,**  здесь это папка Logs и название документа где записаны логи automation.log



automation.log выглядит вот так



Если выполнить framework несколько раз то логи будут идти один за одним, т.е. предыдущее выполнение не будет перезаписано текущим выполнением, получается некая история к которой можно обратится.